International Journal of Computational Linguistics and Applications
vol. 7, no. 1, 2016, pp. 51–65
Received 22/06/2015, accepted 27/07/2015, final 28/09/2015
ISSN 0976-0962, http://ijcla.bahripublications.com

A Micro-Genetic Algorithm for Ontology Class-Hierarchy Construction

SHINYA FUJIHARA[1] AND RAFAEL BATRES[2]

[1] *Suzuki Motor Corporation, Japan*
[2] *Tecnologico de Monterrey, Mexico*

ABSTRACT

Several methods for ontology development have been proposed. However, the development of domain ontologies is still carried out in an ad-hoc manner. This paper explores the use of a micro-genetic algorithm with a seeding scheme based on hierarchical clustering for ontology class hierarchy construction. The micro-genetic algorithm (μGA) is composed of an inner loop and an outer loop. The inner loop consists of: the evaluation of the fitness of each member of the population; the selection of parent chromosomes; the generation of a new population by using crossover and mutation operations; and the separation of the best-fit individual after convergence. The outer loop consists of creating a new random population, transferring the best individual from the inner loop, and restarting the inner loop. The fitness function is based on the correlation between the pair-wise similarities based on the semantic similarity measure of Wu-Palmer and those obtained using Internet and the normalized Google distance (NGD). The proposed approach was tested on the construction of a class hierarchy of machining processes. The results indicate that accurate class hierarchies can be obtained and convergence can be achieved fast with little memory to store the population.

KEYWORDS: ontology construction, class hierarchy, micro-genetic algorithm, normalized Google distance.

1 Introduction

Ontologies are models based on logic that define knowledge structure in terms of classes and subclasses of things and their relationships [1]. Several methods for ontology development have been proposed. Uschold and King [2] propose a methodology that consists of identification of the scope and purpose of ontology, construction of the ontology, evaluation of the ontology and documentation. The methodology application was shown in the domain of enterprise modeling called Enterprise Ontology.

Grüninger and Fox [3] emphasized the use of competency questions to define the requirements as an initial step in the ontology design process. They define competency questions as questions that a knowledge-based system should be able to answer.

Nonetheless, the development of domain ontologies is still carried out in an ad-hoc manner.

Automatic ontology construction has a relatively short history. Khan and Luo [4] propose a method for automatic ontology construction from a set of text documents. The approach assumes that documents that are similar in content are associated with the same concept in the ontology. Firstly, documents are hierarchically arranged using a modified version of the SOTA algorithm [5]. Subsequently, concepts are assigned using WordNet.

Automatic taxonomy generation is a closely related topic which has been reviewed by Krishnapuram and Kummamuru [6]. Specifically, Lawrie et al. [7] propose a graph-theoretic algorithm that generate taxonomies according to a language model. Sánchez [8] proposes an iterative algorithm for hierarchy generation based on terms that are discovered from Internet sources. For example, "solid-state pressure sensor" is found from a search involving "pressure sensor," which in turn is found from a search with the keyword "sensor."

In a previous work, Batres et al. [9] proposed a systematic methodology to design ontologies based on Formal Concept Analysis. The resulting class hierarchy was evaluated using the Internet search combined with the normalized Google distance equation which prompted the question of whether the class hierarchy could be automatically generated using that evaluation method. This paper attempts to answer that question and proposes a micro-genetic-algorithm (μGA) to solve the class-hierarchy generation problem.

This paper is structured as follows. Section 2 describes the proposed methodology. Then, the proposed approach is illustrated with a case study in Section 3. Finally, Section 3 presents the conclusions and discussion.

2 Proposed methodology

We propose the micro-genetic algorithm (μGA) shown in Fig. 1.Typically, μGAs are genetic algorithms with small populations that operate by restarting the population several times while keeping the very best fit individual [10]. Thanks to the small populations, convergence can be achieved faster and less memory is required to store the population. In the proposed methodology, similar to genetic programming, the solution is a tree-like representation.

Generally speaking, a genetic algorithm aims at solving an objective function of the form

$$\max f(\mathbf{x}) \quad \forall \boldsymbol{x} \in X \tag{1}$$

where X is a set of feasible solutions and $f(\mathbf{x})$ is an objective function referred to as fitness. $\mathbf{x}$ is string called *chromosome* or individual composed of elements referred to as *genes*. A genetic algorithm requires a set of n chromosomes known as *population*. Each iteration updates a population using a set of *operators* to lead the next *generation*. The algorithm works by increasing the average fitness of the population.

Specifically, the proposed methodology starts with a set of *initial classes*, which can be obtained through expert consultation and reviews of technical and scientific literature. Alternatively, text and data mining tools can be used to process sources that are stored in an electronic form.

The next step is to generate a random population composed of n class hierarchies, each having the same root class and containing the same set of initial classes. Then the methodology uses the normalized Google distance together with hierarchical clustering to find a good feasible solution, which is inserted to the initial population. Pairs of initial classes are used as keyword terms for Internet search and the number of hits is used for the calculations with the normalized Google distance equation. The distance values are converted to similarity values and stored for subsequent calculations.

The inner loop in Fig. 1 consists of the evaluation of the fitness of each member of the population; the selection of parent chromosomes; the use of crossover and mutation operations to generate a new population; and the separation of the best-fit individual after convergence. The inner loop implements the roulette-wheel scheme [11] for the selection of the parent chromosomes.

The outer loop consists of creating a new random population, transferring the best individual from the inner loop, and restarting the inner loop. In this paper, each cycle in which the inner loop is restarted is called an *epoch*.

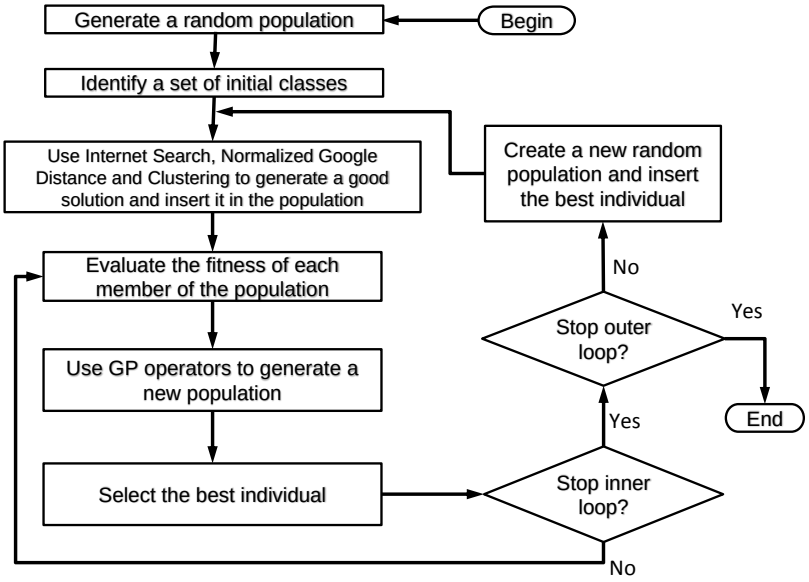

Fig. 1. Proposed methodology

In this paper, the stopping criteria of both loops are given by parameters In_{max} and Out_{max}, which denote the maximum number of iterations of the inner loop and the outer loop, respectively. However, other stopping criteria can be used.

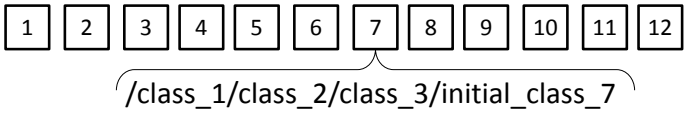

Fig. 2. Chromosome structure. class_1, class_2, and class_3 are newly generated classes and initial_class_7 is an initial class obtained in a preparatory step previous to the class hierarchy generation.

2.1 Solution representation

The solution is encoded as a chromosome that represents the class hierarchy as a tree-like structure. Each gene in the chromosome has an identifier that represents an initial class. Internally, the gene is a list in which the initial classes are positioned based on the depth of the hierarchy. In the example of Fig. 2, gene 7 represents the position of initial class 7. In the internal structure of this gene, the uttermost left element (class_1) represents the root class at depth 1. The second element (class_2) represents a class at depth of 2 that is connected to the root class. The third element (class_3) represents a class at depth of 3 that is connected to class_2 and is the parent class of initial_class_7. As it can be seen from this figure, an identifier is assigned to each class. It is worth mentioning that a maximum depth d_{max} is introduced to control the size of each initial class hierarchy.

2.2 Seeding strategy

A seeding strategy is employed in order to obtain an initial population that leads to better solutions. The seeding strategy consists of incorporating a single good solution to a randomly generated population. The randomly generated population is a population of class hierarchies, each of which contains the initial classes, and classes that are inserted at randomly at selected positions.

The single good solution is obtained by the following procedure:

1. First, the Normalized Google Distance (NGD) [12] is applied to pairwise similarities between all the pairs of initial-classes.

2. Secondly, hierarchical clustering is applied to the distance data obtained in 1.

The NGD distance function is described by equation (2).

$$d(t_i, t_j) = \frac{\max\left(\log f(t_i), \log f(t_j)\right) - \log f(t_i, t_j)}{\log M - \min\left(\log f(t_i), \log f(t_j)\right)} \quad (2)$$

where $f(t_i)$, $f(t_j)$ and $f(t_i, t_j)$ give the number of hits for the terms t_i, t_j and (t_i, t_j) respectively. M is a large number that typically represents the number of indexed documents in a given Web search engine. This paper uses the value of $M=5.8\times10^8$ as reported in [9].

The Web-based similarity is denoted by equation (3).

$$v(t_i, t_j) = 1 - d(t_i, t_j) \quad (3)$$

where $v(t_i, t_j)$ is the Web-similarity of terms t_i, t_j.

2.3 Fitness function

In each solution candidate, we measure the similarity between two classes using the Wu-Palmer similarity measure [13]:

$$sim_{WP}(c_1, c_2) = \frac{2N_3}{N_1 + N_2 + N_3} \quad (4)$$

where N_1 and N_2 are the number of subclass edges from c_1 and c_2 to their closest common ancestor. N_3 is the number of subclass edges from the closest common ancestor to the root class in the class hierarchy.

For n initial classes in C, the Wu-Palmer similarity of each class $c_i \in C$ and each other class $c_j \in C, \forall j = 1 \cdots n$ is evaluated. Subsequently, the Wu-Palmer similarities of c_i are tested for accuracy against the NGD similarities obtained previously for class $c_i \in C$ and each other class $c_j \in C, \forall j = 1 \cdots n$. Testing for accuracy is quantified through the correlation coefficient r_i.

Finally, the fitness function is given by:

$$F = \sum_i^n r_i - \alpha q \tag{5}$$

where αq is a term to avoid bloat in which α is a penalty weight, and q is the number of newly added classes that are not connected to any initial class.

2.4 Genetic operators

Both mutation and crossover are implemented. Mutation is carried out by the *insert*, *remove* and *reconfigure* operators, with probabilities p_{insert}, p_{remove}, and p_{reconf} respectively. The *insert* operator inserts a new class between a randomly selected initial class and its parent class (Fig. 3). The *remove* operator eliminates a class between the top class and an initial class (Fig. 4).

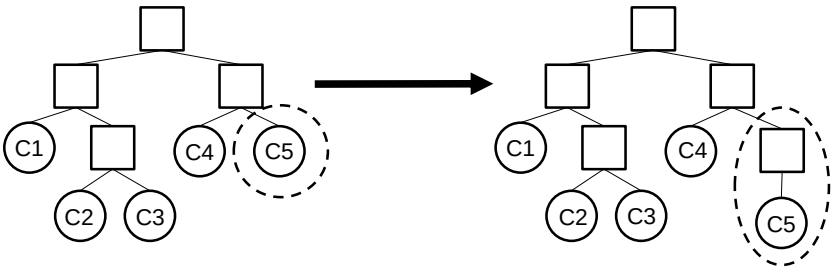

Fig. 3. Mutation insert operation. C1, C2, C3 and C4 are initial classes.

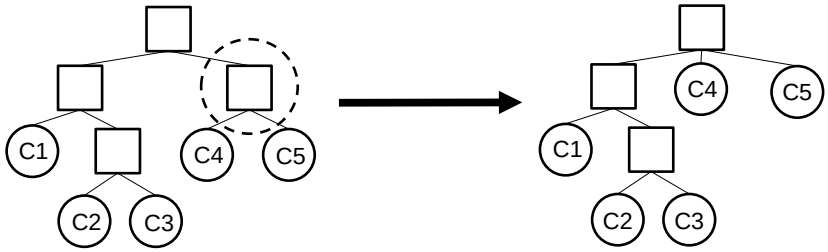

Fig. 4. Mutation remove operation. C1, C2, C3 and C4 are initial classes.

The *reconfigure* operator selects an initial class and modifies the subtree that contains this class (Fig. 5).

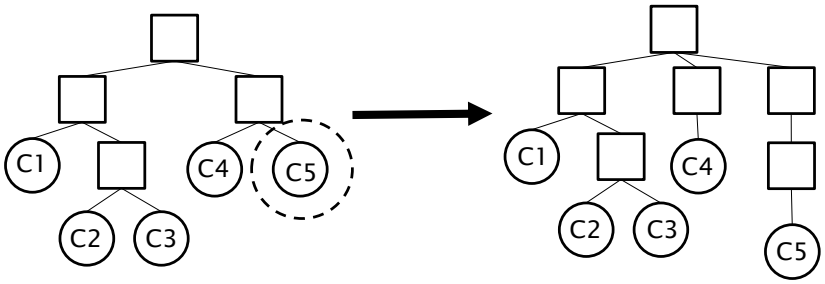

Fig. 5. Reconfigure operation. C1, C2, C3 and C4 are initial classes.

The algorithm performs a crossover operation with probability $p_{crossover}$,. In the crossover operation, two parents are selected from the population to produce two offspring chromosomes (Fig. 6). The crossover operator chooses one or more initial classes and identifies the subtrees located between the chosen initial-classes and the root class in each parent. Subsequently, the selected subtrees are interchanged.

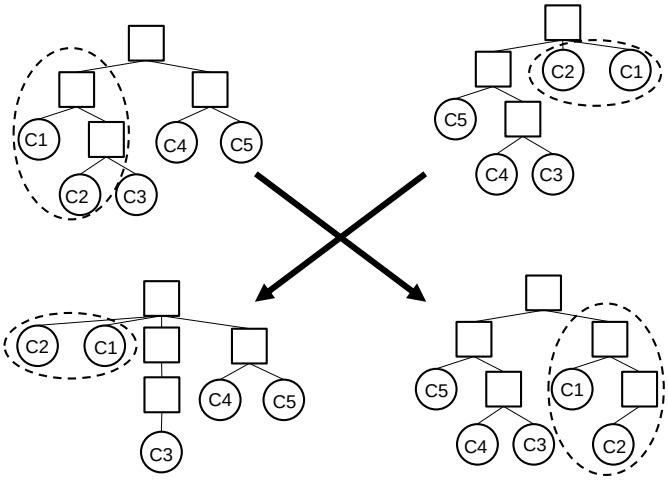

Fig. 6. Crossover operation. C1, C2, C3 and C4 are initial classes.

3 Case study

In this case study, we focus on developing the taxonomy of an ontology for machining processes. Machining processes remove material and modify the surfaces of objects that have usually been produced by other means. For illustration purposes, the scope is limited to mechanical machining. In order to identify the initial classes several common textbooks [14]-[16] and Internet sources were consulted. The initial classes are listed in the first column of Table 1.

In the calculation of the NGD similarities, the Google Scholar search engine was employed. In order to restrict the Web search to the domain of study, keywords were formulated with the inclusion of the term "machining" and search was carried out using double quotes for each initial class. For example, for calculating the similarity between counterboring and spot facing, search with Scholar for "machining" "counterboring" results in f(counterboring)=1019 hits; search for "machining" "spot facing" produces f(spot facing)=620 hits; and search for "machining" "counterboring" "spot facing" results in *f*(counterboring, spot facing) = 56 hits. Substituting these values in Equation (1) gives *d*(counterboring,

spot facing) = 0.2146. Using Equation (2) we obtain v(counterboring, spot facing) = 0.7854. The good feasible solution was obtained by hierarchical clustering, using the normalized Google distances. The correlation coefficients for the comparison between the NGD similarities and the Wu-Palmer similarities for the good feasible solution are shown in Table 1.

A comparison was made between the proposed μGA and a traditional genetic algorithm. Both populations consisted of 48 individuals. The termination criteria was set to a maximum of 4000 generations for the traditional algorithm. The termination criteria for the inner and outer cycle of the μGA were set to a maximum of 100 generations and 40 epochs respectively. The values of these and other parameters used in the experiments are shown in Table 2. The probabilities for all the mutation operators (insert, remove, and reconfigure) were set to 60%.

Table 3 summarizes the results of running the proposed μGA and the traditional genetic algorithm. The results show the proposed approach to be the best algorithm with average performance over ten independent runs of 9.373 and a best fitness value of 9.883 (12 is the maximum value that the fitness function can take for this problem). In comparison, the traditional algorithm produced a class hierarchy with a best fitness value of 8.952. The computation time of the μGA was also slightly better.

Table 1. Correlation coefficients of the initial classes of the hierarchical-clustering generated taxonomy.

	C_i	r_i
Initial classes	reaming	0. 547
	spot facing	0. 594
	counterboring	0. 394
	countersinking	0. 458
	grinding	0. 452
	boring	0. 577
	drilling	0. 702
	milling	0. 568
	turning	0. 313
	tapping	0. 566

blasting	0. 085
lapping	0. 678
$\sum_{i}^{n} r_i$	5. 934

Table 2. Parameters used in the numeric experiments

Parameter	μGA	Traditional GA
Population size	48	48
Iterations of the internal loop	100	4000
Iterations of the external loop	40	–
Maximum depth of the taxonomy	12	12
One-point crossover probability	60	60
Two-point crossover probability	5	5
Mutation probabilities	10	10
Penalty	0. 5	0. 5

Table 3. Results for both methods

Algorithm	Fitness			Average computational time (s)
	best	average	standard deviation	
Traditional GA	8.952	8.850	0.086	45.0
μGA	9.883	9.373	0.325	41.6

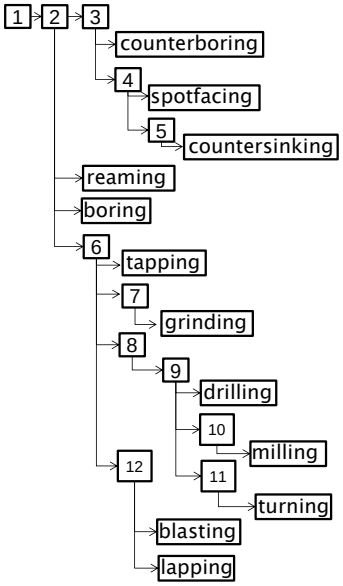

Fig. 7. Class hierarchy obtained with traditional GA.

The class hierarchy obtained with the traditional algorithm is shown in Fig. 7. It can be seen that except reaming, boring, blasting and tapping, few initial classes share the same direct parent class. We would expect grinding, milling and turning to be grouped together because the three machining processes remove materials without requiring the creation of a hole. In spite of these three classes being subclasses of inserted class 6, the classes are too much far apart of each other.

The class hierarchy obtained with the proposed approach is shown in Fig. 8. In this class hierarchy, grinding, milling and turning are grouped together which is consistent with their definition. In addition, it is interesting to note that machining processes that require a previously created hole (reaming, boring and tapping, countersinking, counterboring and spotfacing) are grouped together as subclasses of class 3.

Reaming, boring and tapping which are all direct subclasses of class 3 represent machining processes in which the enlarged portion of the hole is cylindrical. This is in contrast with countersinking, counterboring

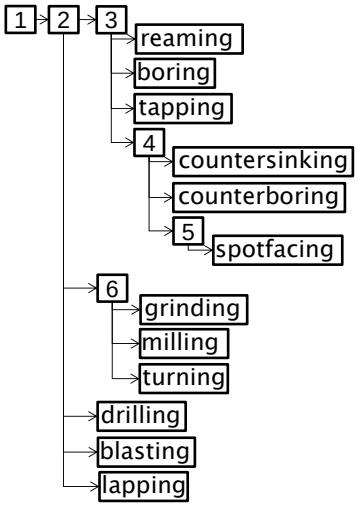

Fig. 8. Class hierarchy obtained with the proposed methodology.

and spotfacing all of which are subclasses of class 4 and produce a bottom part of the enlarged portion of the hole which is either: flat and squared (counterboring and spotfacing) or cone-shaped (countersinking). However, there is no apparent explanation for the grouping of countersinking and counterboring (subclasses of class 4) separate from spotfacing (subclass of class 5).

4 Conclusions

A method has been presented that uses a μGA algorithm to generate ontology class hierarchies. The method is characterized by (1) the use of a good solution that is seeded into the initial population; (2) the use of an evaluation method based on the normalized Google distance and the Wu-Palmer similarity; and (3) the use of small population sizes.

Numeric results show that the proposed method surpasses the traditional genetic algorithm both in terms of quality of the solution

and computational effort. In addition to the advantage in convergence speed, less memory is required to store the population. The class hierarchy also proved to be semantically meaningful when compared with definitions extracted from the literature.

The proposed methodology inserts classes that are unnamed. One challenge is to develop a naming mechanism for these classes. However, the class hierarchies produced with the current methodology can be used for purposes such as semantic similarity calculations [17].

Acknowledgment The authors are grateful to Dr. Yoshiaki Shimizu for his valuable discussions and support of this work.

References

1. Batres, R., Akmal, S.: A Formal Concept Analysis-Based Method for Developing Process Ontologies. Journal of Chemical Engineering of Japan, 46(6), 396–406 (2013)
2. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada (1995)
3. Gruninger, M., Fox, M.S. Methodology for the Design and Evaluation of Ontologies. IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada (1995)
4. Khan, L., Luo, F.: Ontology Construction for Information Selection. Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02) (2002)
5. Dopazo, J., Carazo, J.M.: Phylogenetic reconstruction using an unsupervised growing Neural Network that adopts the topology of a phylogenetic tree. Journal of Molecular Evolution, 44, 222–233 (1997)
6. Krishnapuram, R., Kummamuru, K.: Automatic taxonomy generation: Issues and possibilities. Fuzzy Sets and SystemsIFSA 2003, 184–184 (2003)
7. Lawrie, D., Croft, W. B, Rosenberg, A.: Finding topic words for hierarchical summarization. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, 349–357 (2001)
8. Sánchez, D., Moreno, A.: Automatic Generation of Taxonomies from the WWW. Practical Aspects of Knowledge Management, Lecture Notes in Computer Science, Vol. 3336, 208–219 (2004)
9. Batres, R., Akmal, S.: A Methodology for Developing Manufacturing Process Ontologies. Journal of Japan Industrial Management Association, 64, 303–316 (2013)

10. Krishnakumar, K.: Microgenetic algorithms for stationary and nonstationary function optimization. Proc. SPIE Vol. 1196, Intelligent Control and Adaptive Systems, Guillermo Rodriguez, Ed., 289–296 (1989)
11. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley (1989)
12. Cilibrasi, R.L., Vitanyi, P.M.B.: The Google Similarity Distance. IEEE Trans. Knowledge Data Eng., 19, 370–383 (2007)
13. Wu, Z., Palmer, M.: Verbs semantic and lexical selection. Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics, New Mexico, 133–138 (1994)
14. Nagendra Parashar, B.S., Mittal, R.K.: Elements of Manufacturing Processes, Prentice-Hall of India Private Limited (2007)
15. Degarmo, E.P., Black, J.T., Kohser, R.A.: Materials and Processes in Manufacturing, John Wiley & Sons, New York, USA (2010)
16. M.C. Finishing: Blasting Technical Information, http://mcfinishing.com/resources/blastingtech.pdf
17. Akmal, S., Li-Hsing, S., Batres, R.: Ontology-based similarity for product information retrieval. Computers in Industry, 65(1), 91–107 (2014)

SHINYA FUJIHARA
SUZUKI MOTOR CORPORATION,
TOYOKAWA, JAPAN
E-MAIL: <HUJIHARANE@YAHOO.CO.JP>

RAFAEL BATRES
TECNOLOGICO DE MONTERREY,
CUERNAVACA, MEXICO
E-MAIL: <RAFAEL.BATRES@ITESM.MX>